**Lookup-by-Name ICN仿真平台技术文档**

**李佳伟**

**2019.5.5**

**版本: version 1.0**

目录

[一、整体概览 3](#_Toc8047695)

[二、报文格式介绍 5](#_Toc8047696)

[2.1 请求包 5](#_Toc8047697)

[2.2 数据包 6](#_Toc8047698)

[2.3 注册包 7](#_Toc8047699)

[2.4 解注册包 7](#_Toc8047700)

[2.5 感知消息 8](#_Toc8047701)

[三、程序模块介绍 8](#_Toc8047702)

[3.1 流量生成模块client和Server 8](#_Toc8047703)

[（1）Client 8](#_Toc8047704)

[（2）服务器 9](#_Toc8047705)

[3.2 边界路由器（BRNode）和非边界路由器(NBRNode)介绍 11](#_Toc8047706)

[（1）边界路由器 11](#_Toc8047707)

[（2）非边界路由器 15](#_Toc8047708)

[（3）域内路由forwardingTable 16](#_Toc8047709)

[3.3 资源管理器介绍 17](#_Toc8047710)

[3.4 域间路由模块介绍（PID） 17](#_Toc8047711)

[3.5 NetworkBuilder介绍 18](#_Toc8047712)

[四、使用介绍 20](#_Toc8047713)

[五、总结与展望 20](#_Toc8047714)

# 一、整体概览

本文档描述了一种基于omnetpp的lookup-by-name的ICN仿真平台，为便于描述简称为LICN。其主要调用了omnet自带的函数库、stl库和boost库，在程序设计和编写过程中参考了ccnSim，采用简单模块的设计模式，未考虑底层物理层、数据链路层协议，故未调用Inet的库函数。平台整体概览如图1所示：

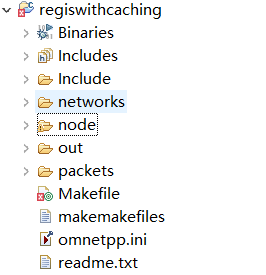


图1 仿真平台整体概览

包含include文件、networks文件、node文件、packets文件。其中include文件包含仿真平台中所有的函数与类的.h头文件，主要包括BRNode.h、clientApp.h、color.h、NBRNode.h、RoutingTable.h、Server.h、zipf.h和zipf\_sample.h如图2所示：

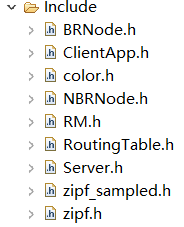


图2 头文件

Networks文件夹存放所有的仿真拓扑。此部分程序可以按照需求，建立相应的拓扑ned文件，目前LICN中只写了较为简单的几个拓扑文件，分别为mesh.ned、Dynamic.ned、RandomGraph.ned、simple\_case.ned。如图3所示：

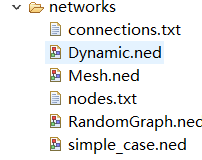


图3 拓扑文件

Node文件夹存放了所有的仿真模块及其对应的程序。主要包含ContentNode文件夹、NetworkBuilder文件夹、RM文件夹、TrafficGenerater文件夹。其中ContentNode文件夹包含：BRNode.cc、NBRNode.cc、RoutingTable.cc、BRNode.ned、INode.ned、NBRNode.ned、RoutingTable.ned。NetworkBuilder文件夹包含netbuilder.cc和netbuilder.ned。RM文件夹包含RM.cc和RM.ned。TrafficGenerater文件夹包含clientApp.cc、Server.cc、clientApp.ned、IApp.ned、Server.ned。如图4所示：

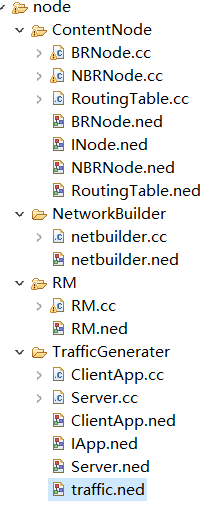


图4 仿真平台主要模块

Packets文件夹中存储了仿真平台中的5种报文格式。主要为请求包、数据包、注册/解注册包、感知包，分别对应get.msg、data.msg、reg.msg、unreg.msg、mon.msg。5中消息对应的枚举类型定义在color.h中。这个文件夹还包含zipf抽样和zipf随机数生成的部分函数，为使得逻辑连贯，后续可以考虑移动到Node/TrafficGenerater文件夹中。

# 二、报文格式介绍

数据报文的格式主要存储与packets文件夹中。报文格式设计没有严格参考color设计。只在其中筛选了使用到的字段。下面分别对其进行介绍。

## 2.1 请求包

请求包格式如图5所示：

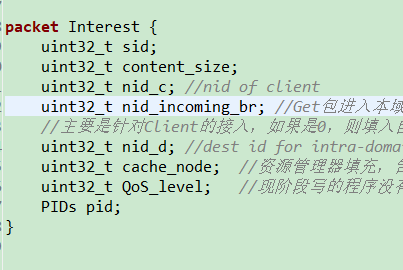


图5 请求包格式

请求包格式包括一下字段sid字段（用来表示请求的内容）、content\_size字段（用来表示内容大小）、nid\_c字段(用来存储请求者的nid)、nid\_incoming\_br字段(用来记录请求者进入本域的边界路由器)，nid\_d字段(用来做请求包的域内路由)、cache\_node字段(用来做内容的域内缓存)、QoS\_level（用来表示请求包服务的优先级，现阶段仿真没有使用这个字段）、pid字段(pid字段为uint32\_t的vector结构，用于存储跨域传输的PID)。

## 2.2 数据包

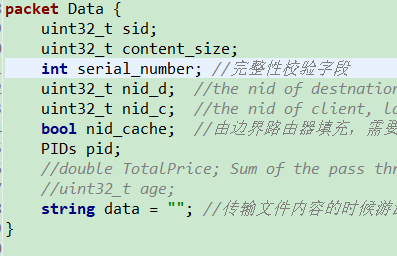


图6 数据包格式

数据包格式如图6所示。主要包含sid字段、Content\_size字段、serial\_number字段（用于做完整性校验）、nid\_d字段、nid\_c字段、nid\_cache字段（有边界路由器填充，如果需要在本域缓存，填充为true，不需要则填充为false）、pid字段、data字段（用于在传输问件时，封装文件数据）。此外，还有两个字段没有使用，分别为Total\_price（可以用来统计客户端到内容源之间路径的所有价格）、age字段（可以用来统计信息年龄）。

## 2.3 注册包

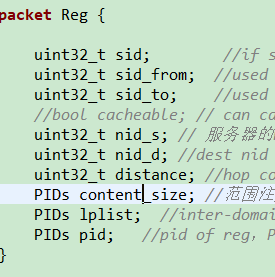


图7 注册包格式

注册包格式如图7所示。包含sid字段(sid如果为0，表示范围注册)、sid\_from字段（范围注册）、sid\_to（范围注册）、nid\_s(服务器nid、跨域注册时填充AS号)、nid\_d(域内转发的目的nid)、distance(存储以AS为单位的跳数)、content\_size（uint32\_t的vector，存储范围注册时内容大小的序列）、lplist（域间路径喜好度序列）、pid（PID序列）。

## 2.4 解注册包

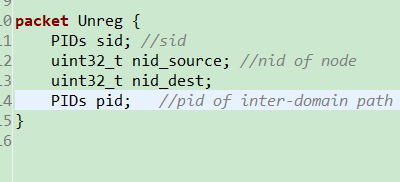


图8 接注册包

解注册包格式如图8。包含sid字段(与pid序列一样，用uint32\_t的sid字段存储，可以同时解注册多个内容)、nid\_dource字段（解注册节点的nid，跨域为AS号）、pid。

## 2.5 感知消息

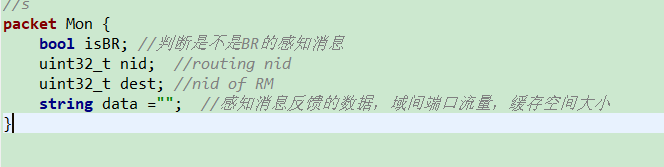


图9 感知消息格式

感知消息字段如图9所示。包含isBR字段(判断是BR还是NBR)、nid字段（发送感知消息的节点nid）、dest字段（填写RM的nid）、data（存储感知消息，反馈域间路径流量大小和缓存空间利用率等，BR和NBR此处封装的数据不同。）

# 三、程序模块介绍

## 3.1 流量生成模块client和Server

Client和Server用于生成仿真中的流量。Client用于生成get包，Server用于生成注册包和返回data包。

### （1）Client

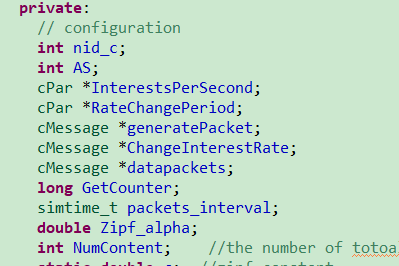


图10 client参数

Client主要包括一下参数InterestsPerSecond、RateChangePeriod、GetCounter、pacets\_interval、zipf\_alpha、NumContent、c。其中InterestPerSecond字段用于从cleint.ned中读取每秒钟client生成的请求包数目，利用请求包每秒的个数和负指数分布，可以算出请求包的发送间隔，如图下图公式所示：



图11 泊松请求包发送间隔

图11所示的公式，用于模拟泊松过程的请求包达到。此外也可以设置其他不同形式的请求包发送间隔如均匀分布、固定值等，如图12所示：



图12 均匀分布和定长的发送间隔

通过对client.ned中的ChangeRate字段进行配置（0或1）。可以在仿真进行过程中，改变节点的请求包发送频率，具体为改变”InterestsPerSecond”的数值。

客户端按照zipf，封装get包sid。从而内容流行度也服从zipf分布。相关函数为unsigned int clientApp::zipf（doiuble alpha, unsigned int n）。alpha为zipf的参数，n为仿真过程中设置的内容总数。

### （2）服务器

服务器端参数如图13所示：

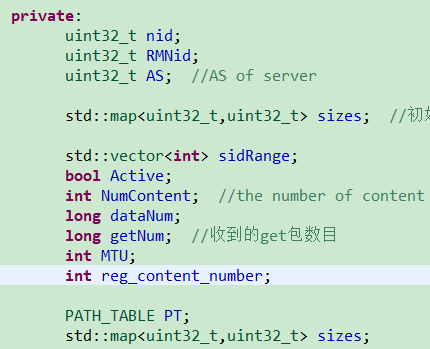


图13 服务器参数

服务器初始化包括两个阶段。第一个阶段，初始化服务器的配置，具体为初始化每个模块的forwardingTable。第二个阶段，服务器才能针对其提供的内容，进行内容注册，服务器初始化不能放到第一阶段进行，因为第一阶段网络的路由器需要初始化forwardingtable，有forwardingTable后，才能将注册包发送到资源管理器。图13中的参数，MTU表示单个数据包的长度，reg\_content\_number表示本服务器提高的内容各数，NumContent表示仿真的内容总数，dataNum用于统计发送的data包个数，getNum用于统计发送的get包个数。

图13所示的参数中，sidRange参数用于从server.ned文件读取服务器提供的内容的sid。通过range，可以计算出每个服务器可以提供的内容个数。部分程序如图14所示：

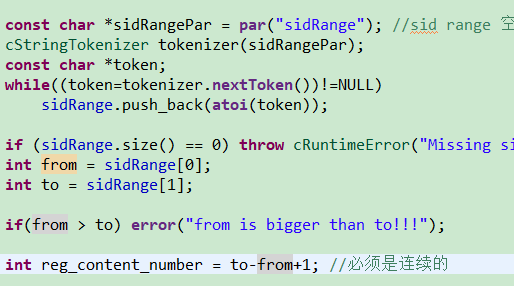


图14 计算服务器维护的内容个数

服务器作为内容提供者，初始化内容大小。内容大小可以采用集合分布、均匀分布、固定值等各种分布，根据文献来看，均匀分布、固定值都是在文献中出现过的假设。如图15所示。

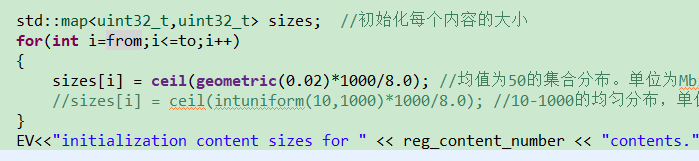


图15 内容大小服从均值为50Mbits的几何分布

服务器收到请求包后，会返回相应的内容长度的内容。按照MTU来封装data包。如以太网中MTU为1500。

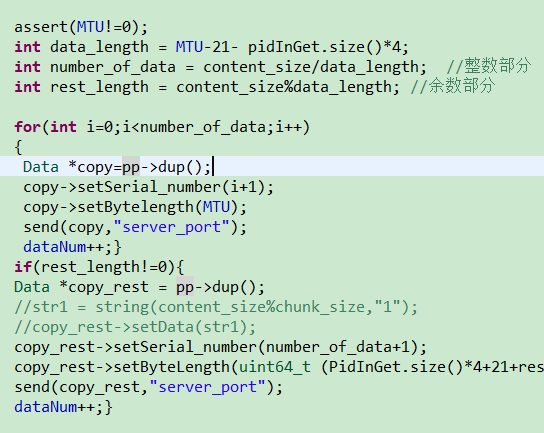


图16 服务器按照MTU封装data包

setByteLength可以直接控制data包的长度，单位为字节。如果需要服务器返回有意义的数据，例如文件、视频等，则在data包的string data字段（见图6），封装内容数据即可。

## 3.2 边界路由器（BRNode）和非边界路由器(NBRNode)介绍

边界路由器和非边界路由器为仿真程序的主要模块。用于组件网络拓扑。下面分别进行说明。

### （1）边界路由器

边界路由器功能比域内路由器复杂。主要参数如图17所示：

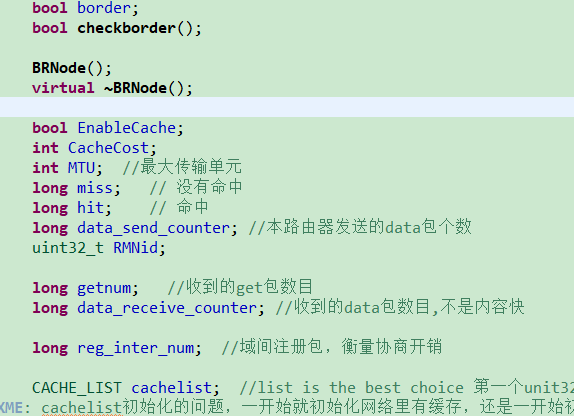


图17 边界路由器参数

AS参数为边界路由器所在的AS号，用于在网络初始化的时候，以AS为单位，构建网络的域内路由表，不同的AS号是一个不同的域。Nid\_br是本边界路由器的nid，表示此边界路由器的身份，用来做域内路由的目的地址。Cachespace表示路由器的缓存缓存空间。Avaliable\_cache\_space表示本节点的可用缓存空间。Replacementpolicy用来设置本节点的缓存空间，replacementpolicy的取值如图18所示。PendingCacheTable用来判断收到data包是否需要在本域缓存此内容。PacketCounter\_Bytes 用来统计字节单位的域间流量，packetCounter\_packets用来统计包单位的域间流量。ForwardingTable用来存储域内转发的路由信息，维护了目的地址到转发端口和距离的映射。PT表存储了所有域间PID的信息，包含PID对应的两个nid，两个端口号，和所述的AS号。PidsOnThisBR存储了本节点的域间PID，如果本BR没有PID，则为空。Border和isBorder()函数结合，用于判断是否是BR。EnableCache用于配置本节点是否开启缓存功能。CacheCost用于设置本节点的缓存开销。MTU为最大传输单元，当命中缓存时，封装MTU长度的data包返回内容。Miss统计本节点所有没有命中的get包个数。Hit统计本节点命中缓存的Get包个数。Data\_send\_counter统计本节点发出的data包个数（提供的缓存内容），不是转发的data包个数。getNum为本节点收到的所有Get包个数，data\_receive\_counter为收到的data包个数。Reg\_inter\_num为本节点收到的所有域间注册包的个数。Cachelist为本节点的缓存列表。

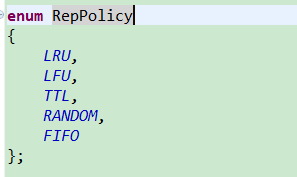


图18 缓存替换策略

边界路由器的初始化也是分两个阶段进行。第一个阶段为初始化众多边界路由器的参数，包含缓存替换策略，网络拓扑的域内路由等。第二个阶段没有任何操作

边界路由器的消息处理，根据收到的消息类型，进行不同的处理流程。主要判断：（1）networkstates消息，发送感知消息给RM，包括本节点的域间路径的流量，缓存空间利用情况；（2）GET，进入get处理逻辑，提取sid，查找缓存列表，有则返回内容，无则判断目的地址，转发到RM或者发往域间。（3）data包，判断域间来的还是域内来的，然后对尾部的PID进行删除操作；查询pendingCacheTable，看是否需要在本域缓存内容，需要则转发到缓存节点，不需要则根据pid看发往域间，还是发往本地的客户端。如果自己就是缓存节点，则进行缓存的操作，然后在判断是发往域间还是发达本地客户。（上述逻辑没有按照代码逻辑写，具体参见程序逻辑，会更加清晰）。（4）注册包，判断域间发来的还是域内来的，域内来的是缓存发的注册，域间来的是别的AS发来的注册包，RM上维护的distance需要通过注册包的distance来计算距离。发往域间的注册包源地址写AS号，本地发往RM的注册包，填写缓存节点的nid。从而在RM上的注册表，可以创建相应的内容可达性条目。（5）解注册包，类似注册包的处理流程。在封装相应字段上有一些区别。（6）monitoring包，感知包不是networkstates消息，前者是selfmessage，用来周期性驱动给RM汇报本节点情况。Mornitoring包是别的节点发到RM的汇报消息，直接转发到本地RM即可。

边界路由器有几个重要函数，如图19所示：

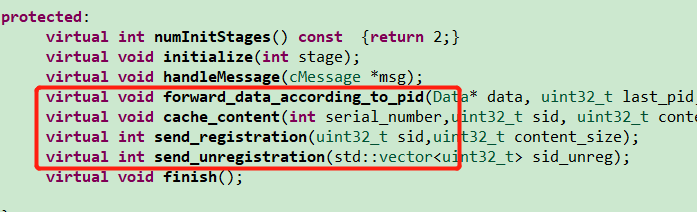
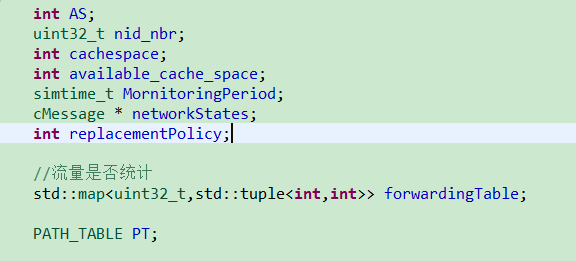


图19 边界路由器重要函数

Forwarding\_data\_according\_to\_pid，用来根据pid转发data包。Cache\_content用来缓存。Send\_registration用来发送注册包。Send\_unregistration用来发送解注册包。注册包一次只能注册一个sid。解注册包可以一次解注册多个sid。因为缓存替换时，不同内容大小不同，一个siid对应一个内容，因此为存储一个大的内容，可能删除多个小内容的缓存，从而导致需要解注册多个sid。

### （2）非边界路由器

非边界路由器的设计与边界路由器类似。参数上也有部分重合。例如缓存替换策略，如图20所示。



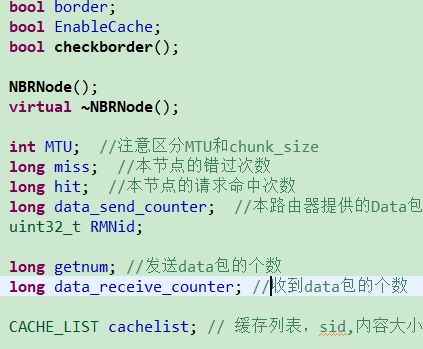


图20 NBR参数说明

NBR的参数与BR的参数类似，重复部分不加以说明。NBR初始化也分两个阶段。第一阶段初始化各种参数以及构筑域内路由表。第二个阶段没有操作。

消息处理也类似BR，分为networkstates，get，data，reg，unreg，mornitoring。Networkstates用来周期性汇报NBR的状态给RM，Get包到达NBR后，也需要查找缓存是否命中，是的话则返回data，不是的话则发往本地RM。Data包没有pendcahcetable的设计，通过目的地址判断是否需要缓存，如果目的地址是自己且尾部pid不是自己，则需要缓存，如果目的地址不是自己则正常转发即可。Reg、unreg、mornitoring消息都是直接判断转发。

NBR也有类似于BR的三个函数，forward\_data\_according\_to\_pid，cache\_content，send\_registration, send\_unregistration。这里不在赘述。

### （3）域内路由forwardingTable

域内路由通过BR和NBR上都有的一个于内转发表forwardingTable来实现。forwardingTable的构建，采用了最短路图轮算法里的最短路径算法。距离的单位为源目的节点之间的跳数。关键函数为

calculateUnweightedSingleShortestPathsTo()函数。



图21 无权重最短路函数

后续可以在此基础上扩展，修改为带权重的最短路，包括路径权重和节点权重。相关接口为Node类里的wgt和link类里的wg，如图22所示：

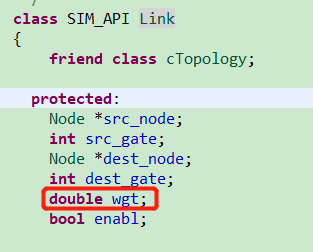
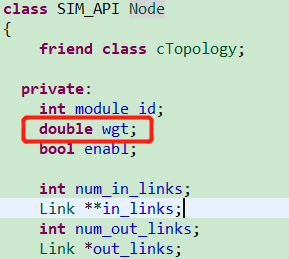


图22 Node类和link类中的权重接口

当使用有权重的拓扑，构建网络时，可根据Dijkstra权重最短路算法，生成forwardingTable，相关接口函数为：

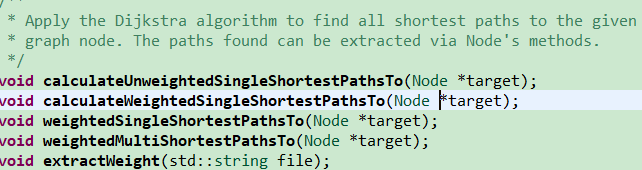


图23 生成forwardingTable的最短路算法函数

## 3.3 资源管理器介绍

资源管理器是仿真中的重要模块，主要参数如图24所示：



图24 RM主要参数列表

AS号表示RM所属于的AS，nid\_rm记录rm的nid，cachenode\_info存本域可以缓存的节点以及总空间、可用空间、缓存成本(cache cost)等信息，Inter\_traffic\_packets存储本域域间PID及其感知到的对应的流量信息（单位为包）、Inter\_traffic\_bytes存储本域域间PID感知到的字节单位的流量，decision\_where\_policy决定缓存的位置的策略。Decision\_where\_policy和decision\_which\_policy的宏定义如图25所示：

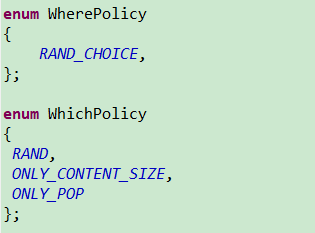


图25 缓存策略的宏定义

如图25所示，现在缓存位置的决定策略，只写了简单的随机选择（RAND\_CHOICE）域内缓存节点的方法，后续依据优化目标，可以开发更多的缓存放置算法。缓存内容的选择算法目前写了3中简单的策略，为RAND、ONLY\_CONTENT\_SIZE、ONLY\_POP策略，分别表示随机选择缓存内容，根据内容大小判断是否缓存、根据内容流行度判断是否缓存，内容流行度的定义为请求的次数除以仿真的时间。Nego\_size\_threshold为协商的门限值，与内容的大小有关，当域间传输的内容大于门限时，才对此内容进行协商注册。Nego\_pop\_threshold类似，当内容流行度大于此门限值时，猜对内容注册进行协商。DestAS\_To\_Pids表存储了AS号到PID的映射，其意义是邻居AS和到达邻居的所有的域间PID，在域间转发兴趣包和注册包是，RM需要查询此表。RegTable是注册表，存储了服务标识到{<nid/AS，内容距离，请求频率/次数>,<协商的域间路径选择概率>}，forwardingTable用与域内转发的路由表，stablilization\_time为假设的仿真系统稳定时的时间，PT存储域间的PID的信息。

RM初始化阶段需要读取ned的配置信息，初始化forwardingTable。随后处理其收到的消息，依据消息类型调用不同的处理函数，分别为handleInterest()，handleReg()，handleUnreg()，handleMon()。此外，还有几个重要的函数，calculate\_poreference\_to\_pids()函数用于计算域间传输路径的喜好度。Return\_nid\_by\_uisng\_pid\_and\_AS()通过提供pid来返回此PID所对应的本域的NID。计算域间路径喜好度有三种方式，如图26所示：

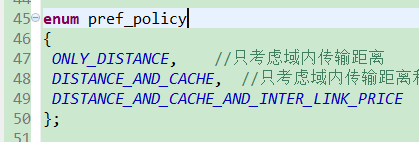
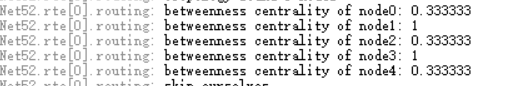
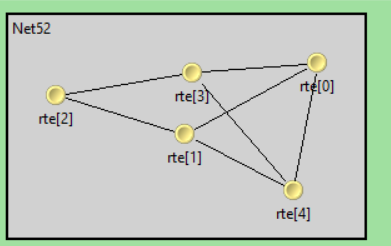


图26 计算路径喜好度的三种策略

计算路径喜好度的三种策略，ONLY\_DISTANCE方式为根据域内的内容传输距离来计算域间路径的喜好度。DISTANCE\_AND\_CACHE只考虑域内传输距离和节点的缓存情况，来计算域间的喜好度，在考虑域间传输距离的同时，也考虑每个节点的缓存情况。DISTANCE\_AND\_CACHE\_AND\_INTER\_LINK\_PRICE策略，在策略2的基础上，额外考虑了域间路径的传输价格，这种测量可以用来模拟域间是Peer关系，还是Provider和Customer的关系。

后期可以添加基于中介中心度的数值选择缓存节点。调用下面的函数即可。



图x 中介中心度

上图中的中介中心度为0.33,1,0.33,1,0.33。

## 3.4 域间路由模块介绍

路由模块用于构筑几个重要的表，供仿真的所有模块访问使用，如图22所示：

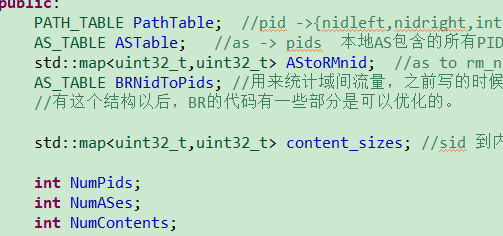


图22 路由模块设计

路由模块有pathTable表、ASTable表、AStoRMnid映射、BRNidToPids映射、content\_sizes表。PathTable表提供了PID到两个nid，两个端口号，两个AS号的映射。ASTable提供了AS号到本AS所有Pid的映射。AStoRMnid提供了本AS到本域RM的映射，用于帮助仿真中的每个节点直达本域的RM的nid。BRNidToPids提供了边界路由器到本边界路由器所有Pid的映射。Content\_sizes存储了sid 到内容大小的映射。NumPids为网路中的pid 总数，numASes为网络的AS总数，numcontents为内容总数。

下面分别描述PathTable、ASTable、AStoRMnid映射表、BRNidToPids映射表、content\_size表的生成方式。

pathTable表，在初始化阶段，程序通过识别BR节点，生成由BR节点构成的网络拓扑。此时拓扑中的路径，可以是域间路径，也可以是域内路径，区别在于路径两端的BR是不是属于同一AS。做出相应的判断，识别出域间路径后，为他们分配uint32\_t类型的PID号，识别PID对应的BR的NID号，BR所属的AS号，以及PID在BR上对应的端口Interface的编号。然后初始化并存入上述的PathTable、ASTable、AStoRMnid映射表、BRNidToPids映射表。

为了保证在整个仿真过程中，每个SID对应的内容大小不发生改变，需要建立维护sid到Content\_size的表。SID对应的内容大小，一旦通过均匀分布、几何分布等随机数生成方式生成，就会被存入这个表中。从而确保内容大小在整个仿真过程中的一致性。

## 3.5 NetworkBuilder介绍

网络拓扑可以通过ned文件自己手动编写，但是当规模较大的时候，就存在不方便的问题。这时需要可以通过给定nodes.txt和connections.txt两个文件，来让程序读取文件数据，自动构建仿真用的网络拓扑，从而当节点数目较多以及网络拓扑复杂时，省去繁琐的ned编写工作。

图24为node.txt的文件举例，第一列为节点的id，对应仿真中的nid，第二列为节点的名字，用处不大，删掉也无所谓，第三列为节点的模块类型：

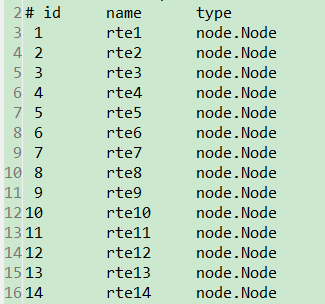


图24 node文件举例

图25为connection文件举例，第一、二列为源目的节点对，第三列为时延、第四列为误码率、第五列为链路带宽：

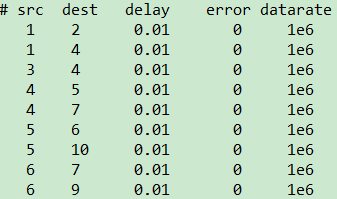


图25 connection文件举例

通过给定上述两个文件，NetworkBuilder可以自己构建一个网络拓扑。

# 四、使用介绍

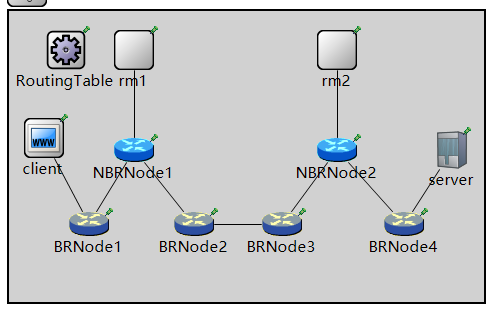


图26 简单的拓扑举例

简单的拓扑如上图所示，网络中客户端和服务器必须连接在BR节点上，不能连接到NBR节点上，除此之外没有别的要求。上图中Client，BRNode1，BRNode2，NBRNode1，rm1属于一个AS；rm2，NBRNode2，BRNode3，BRNode3，Server属于另外一个AS。节点的Nid和所属AS，也需要在拓扑ned文件中给定，如图27所示：

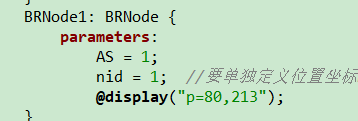


图27 必要参数初始化

大规模或者更多场景的网络拓扑，可以通过使用NetworkBuilder和配置仿真ini文件来实现。

# 五、总结与展望

本文档主要介绍了一种lookup-by-name的ICN仿真平台的设计与实现。当前版本的Lookup-by-Name的ICN仿真平台，与ccnSim的仿真平台设计思路类似。具有轻量级(light-weight)、扩展性强、不依赖inet库等优点。但是，本平台也存在一些问题，总结有一下几点：

1. 报文格式设计不够完善

报文格式目前只设计了部分功能字段，如若实现更多的功能，例如域间pid改变的协商功能，还需要添加相应的字段和程序。

2. 客户端或缓存节点缺少完整性校验程序以及可靠传输协议

客户端和缓存节点缺少完整性校验程序，缺少丢包重传等可靠传输协议。

3. 缓存策略有待完善

目前，只编写了几种缓存策略，更多的策略需要后续的补充开发。

4. 代码效率、冗余变量可进一步优化

部分代码重复性的可以重构为函数。部分代码可以优化，从而提高效率，例如CacheList中的Content\_size，因为Get包和Data包中都封装了Content\_size字段，所以其实CacheList里可以不存内容大小的数据。再比如RM上的Inter\_Traffic\_packets表完全可以和inter\_traffic\_bytes表合二为一，从而节约程序的内存空间，提升程序的效率。上述类似的问题应该还有很多，如果想让这个平台更好更完善，还得需要细致的“打磨”。

上述问题可以后续迭代开发进一步完善。